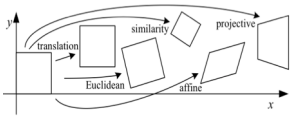
Surface orientation: color, texture, and reflectance properties (Lambertian vs specular)

Flood Fill: Depth-first Traversal (self)
– If self marked, stop!
– Else!
• Mark self !
• For each neighbor!
– Depth-first traversal (neighbor)

Computer Vision
– How are images represented?
– How are images formed?
– What kinds of tasks are we interested in doing?
- How do you find what you are looking for?
Morphological operators, Binary image analysis, gradients, general filter responses (mask, kernel, integral image), corners and key points

Different points of view about tasks in computer vision: algorithmic, signal processing, and machine learning

SIFT - detect and describe local features in images: corners and dots
Scale Space Octaves - Every time the width of your Gaussian doubles, down sample the image
Gradient Orientation Histogram
• Make a histogram over gradient orientation
• Weighted by gradient magnitude
• Weighted by distance to key point
• Contribution to bins with linear interpolation
Computing SIFT Descriptor
• Divide 16 x 16 region surrounding keypoint into 4 x 4 windows
• For each window, compute a histogram with 8 bins
• 128 total elements
• Interpolation to improve stability (over orientation and over distance to boundary of window)



$$T(t_x, t_y) = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(\alpha) = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\theta) = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation: Projective
• Also called: Perspective.
Most general linear transformation
• What changes:
– The overall location
– Orientation wrt coordinate axes
– Distances
– Angles between vectors
– Parallel lines no longer parallel
• What stays the same:
– Straight lines stay straight

Estimating a Linear Transform

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

• 6 coefficients for an affine transforma3on
-> 6 rows of matrix
-> 3 correspondences needed (x and y give one row each)

Multiple Hypothesis Tracking
• Build out the MHT trees
• Set up corresponding matrices
• Score each leaf (using a Kalman filter)

What can you do with optical flow?
• Infer the motion of the camera
• Calculate when your robot is going to crash
• Infer structure of a scene
• Motion segmentation
• Video stabilization
Goal
• Given: two images separated by small dt
• For every pixel in an image, find a vector (u,v) representing the motion of the pixel
i.e. arrow pix 1st image to pix 2nd image

Classification with Adaboost
• Training:
– Given a pool of "weak learners" and some data,
– Create a "boosted classifier" by
– choosing a good combination of K weak learners and associated weights

• In our case "Train a Weak Learner" == Choose a feature to use and which threshold to apply

Using Features for Classification
How do you know which feature to use?
-Try them all and pick the one that gives the best result
-Then, choose the next one that does the next best job, emphasizing misclassified images

• Each threshold on a single feature gives mediocre results, but if you combine them in a clever way, you can get good results
-That's the extremely short version of "boosting"

Classification Cascade
• Solution: Use a "cascade" of increasingly complex classifiers
• Create less complex classifiers with fewer weak learners that achieve high detection rates (maybe with extra false positives)
• Evaluate more complex, more picky, classifiers only after the image passes the early classifiers
• Train later classifiers in the cascade using only images that pass earlier classifiers

Properties of Image Regions
• "Location"
– Set of Pixels
– Boundary
– Centroid (center)
– Bounding Box
– Oriented Bounding Box
– Convex Hull
• "Shape"
– Area (Total pixels)
– Perimeter (Total pixels in boundary)
– Orientation (Axis of least inertia)
– Elongation (e.g. ratio of dimensions of oriented bounding box)
– Compactness (e.g. ratio of region area to bounding box size)
– Circularity (e.g. ratio between perimeter^2 and area
• "Appearance"
– Mean brightness
– Brightness variance
– Brightness histogram
– Mean color
– Color histogram
– Texture (some statistics about gradients)

• Jaccard Index (aka – Precision, Overlap Ratio)
• A = all pixels in object A
• B = all pixels in object B
• Example:
|A inter B|/ |A ∪ B|

Brightness Statistics: Count, mean, variance
Otsu's Method of automatic thresholding: Choose a threshold so that the sum of variance of image values in foreground and background regions is minimized

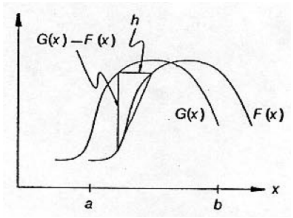
Segmentation
Region Growing (Merging)
Initialization = all connected components with shared pixel value
• Grow Criteria = merge regions/pixels when grey-level value intensity between adjacent regions is at most 1.

Region Growing (Watershed)
Initialization = all minimum values
• Grow Criteria = increment each seed region pixel value and add to it any pixels that are less than or equal to its pixel value; create dam when two regions merge
• Stop = when all pixels are assigned to a region
Note: dam points occur when dilation causes regions to merge

RANSAC - RANdom Sample And Consensus
• Robust estimation of parameters in the presence of outliers mistakes & noise)
• Given: Corresponding points, error threshold, number of iterations
• In a loop – Choose a sample of the points
– Estimate the parameters
– Compute the error
– Count number of inliers (error < threshold)
– Keep track of the sample and estimated parameters with most inliers and/or lowest error
• How many trials (S)?
– How many points to estimate parameters? (k)
– What is inlier probability? (p)
– What probability that you find a set of inliers? (P)
– Probability that all points in a trial are inliers? p^k
– Probability that after S trials you will not have found a sample with all inliers? (1 – p^k) ^ S
– Probability of finding a set of inliers that you want? P
– (1 – P) = (1 – p^k)^S
– S = log(1-P) / log(1 – p^k)

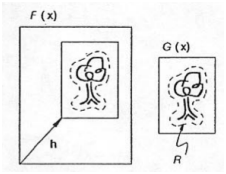
$$E = \int (\alpha(s)E_{\text{cont}} + \beta(s)E_{\text{curv}} + \gamma(s)E_{\text{image}})\, ds$$

Combining Images
The Pipeline
Image Sequence
Localize Features - Image Keypoints
Extract Features
Match Features - Features
Estimate Transformation - Correspondence Pairs
Transformations
Choose Frame of Reference
Transform Image - Transformed Image
Transfor Weight Map - Error Image
Find Boundary - Blending Boundary
Combine Images
Mosaic
Three Type: "Copy and Paste", Distance Transform – Use the pixels from the source image whose center is closest to the output pixel, and Multi-band / Pyramid blending (convolve image with successively larger Gaussians DoG, Minimum Error Boundary



Types of Tracking: Tracking by detection, Optical Flow/Dense scene motion, Contour Tracking, Multi-target tracking

Template Tracking
- Given: small image patch of something we're looking for
- Goal: Find the best-match location in the new image
- How: Search in a small window around its previous location
- How to compute a matching score?
- Normalized Correlation Coefficient
- Given: template, initial location x0
- For each image, t=1:N
  - Search in a small window around x_{t-1}
  - x_t is location with highest NCC score
- Challenges:
  - Computational Cost
  - Getting lost / Drift
  - Non-translational motion (e.g. rotation)
  - Non-rigid motion (articulation of hand)
  - tiD motion
  - Changing appearance of real object
- What are the benefits/downsides of using larger templates/search windows?
- Why is rotation and scaling problematic for a template tracker?
- If we update the template as we track, what problems do we solve or create?
- How could we benefit from using a constellation of smaller templates instead of one big one?



Background Modeling
-Use first frame, occasionally reinitialize
-Compute the mean of observed pixels over time, running average, "statistical model", dynamically update to compensate for uninteresting changes

$$\mu_t = \alpha * x + (1-\alpha)\mu_{t-1}$$

-Multi-modal distribution: Mixture of Gaussians MoG

Horn-Schunck - algorithm assumes smoothness in the flow over the whole image. Thus, it tries to minimize distortions in flow and prefers solutions, which show more smoothness. Constant brightness assumption, Lambertian surface and light doesn't change
- Smoothness
  - The optical flow is a vector field
  - Each pixel has a flow vector (u,v)
  - The vectors in the flow field should vary smoothly
  - Magnitude of gradient of flow vector components should be small

Objective Function

$\mathscr{E}_b = E_x u + E_y v + E_t$

measure of the departure from smoo

$$\mathscr{E}_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

$$\mathscr{E}^2 = \int\int (\alpha^2 \mathscr{E}_c^2 + \mathscr{E}_b^2)\, dx\, dy.$$

- Smoothness
  - The optical flow is a vector field
  - Each pixel has a flow vector (u,v)
  - The vectors in the flow field should vary smoothly
  - Magnitude of gradient of flow vector components should be small

Lucas-Kanade (Sparse Optical Flow)
-Assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration
-Since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image.

$F(x+h) = F(x) + hF'(x)$ Taylor and Newton-Raphson find roots

- Two curves: F(x), G(x)
- Displacement: h
- Goal: Find the displacement
- (Derivation on board)
- Assume: G is a translated version of F: G(x) = F(x+h)
- Assume: First-order approximation is sufficient F(x+h) = F(x) + h F'(x)
- Assume: Displacement is small
- Algorithm:
  - For each patch, use previous location as initial guess
  - Until error (F(x+h)–G(x)) is sufficiently small
    - Compute the summations over the gradient
    - Compute the summation over the image values
    - Find displacement, h, by:

$$h = \sum_x \nabla F(x)\,[G(x) - F(x)] \left[\sum_x \nabla F(x) \nabla F(x)^T\right]^{-1}$$

  - Use equations to guide search over several iterations
- Details: – Compute sums with weights
- distance to center
- mitigate regions where image values match but gradients do not
  - When misregistration might be large wrt image patch
    - Smooth image
    - Coarse-to-fine strategy (search in low resolution image for approximate match, then refine in high resolution image)

Kanade–Lucas–Tomasi - less costly KLT makes use of spatial intensity information to direct the search for the position that yields the best match.
- What properties of this thing can we use?
  - If the matrix is not invertible, we can't track this patch
  - How do we know if it is invertible? Its determinant
  - Slightly more information: eigenvalues
- Find regions of the image where
  - Eigenvalues are "sufficiently large" (larger than in a patch that is just noise)
  - Ratio of eigenvalues is "reasonable" (matrix is well-conditioned)

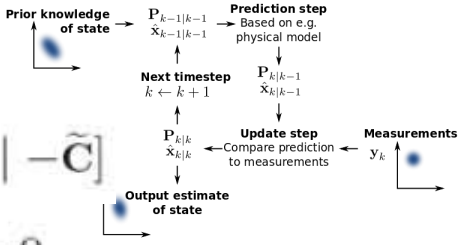
Shi-Tomasi Use translation for tracking and affine model for deciding when to give up

Predict [edit]

Predicted (*a priori*) state estimate	$\hat{\mathbf{x}}_{k\mid k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1\mid k-1} + \mathbf{B}_k\mathbf{u}_k$
Predicted (*a priori*) estimate covariance	$\mathbf{P}_{k\mid k-1} = \mathbf{F}_k\mathbf{P}_{k-1\mid k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$

Update [edit]

Innovation or measurement residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k\mid k-1}$
Innovation (or residual) covariance	$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k\mid k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \mathbf{P}_{k\mid k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\hat{\mathbf{x}}_{k\mid k} = \hat{\mathbf{x}}_{k\mid k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\mid k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k\mid k-1}$



$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\tilde{\mathbf{C}}]$$

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0.$$

- F is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If x and x′ are corresponding image points, then $\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - $\mathbf{l}' = \mathbf{F}\mathbf{x}$ is the epipolar line corresponding to x.
  - $\mathbf{l} = \mathbf{F}^{\mathrm{T}}\mathbf{x}'$ is the epipolar line corresponding to x′.
- **Epipoles**:
  - $\mathbf{F}\mathbf{e} = \mathbf{0}$.
  - $\mathbf{F}^{\mathrm{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** P, P′:
  - General cameras, $\mathbf{F} = [\mathbf{e}']_\times \mathbf{P}'\mathbf{P}^+$, where $\mathbf{P}^+$ is the pseudo-inverse of P, and $\mathbf{e}' = \mathbf{P}'\mathbf{C}$, with $\mathbf{P}\mathbf{C} = \mathbf{0}$.
  - Canonical cameras, $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$, $\mathbf{P}' = [\mathbf{M} \mid \mathbf{m}]$, $\mathbf{F} = [\mathbf{e}']_\times\mathbf{M} = \mathbf{M}^{-\mathrm{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathbf{M}^{-1}\mathbf{m}$.
  - Cameras not at infinity $\mathbf{P} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]$, $\mathbf{P}' = \mathbf{K}'[\mathbf{R} \mid \mathbf{t}]$, $\mathbf{F} = \mathbf{K}'^{-\mathrm{T}}[\mathbf{t}]_\times\mathbf{R}\mathbf{K}^{-1} = [\mathbf{K}'\mathbf{t}]_\times\mathbf{K}'\mathbf{R}\mathbf{K}^{-1} = \mathbf{K}'^{-\mathrm{T}}\mathbf{R}\mathbf{K}^{\mathrm{T}}[\mathbf{K}\mathbf{R}^{\mathrm{T}}\mathbf{t}]_\times$.

What can you do with camera geometry?
- Given 3D objects, find their image coordinates
- Given image points and tiD points, infer the pose of the camera
- Given image coordinates and camera matrices for multiple cameras, reconstruct the 3D point
- Perform multiple view data association for tracking
- Infer the movement of the camera (from image correspondences only)